Computing and Visualization in Science

Real time numerical simulation and visualization of electrochemical drilling

M.J. Noot, A.C. Telea, J.K.M. Jansen, R.M.M. Mattheij

Eindhoven University of Technology, Dept. of Mathematics and Computing Science, NL-5600 MB Eindhoven, The Netherlands

Abstract. Gas turbines have to be provided with holes in order to provide cooling; these holes are made using an electrochemical drilling technique. Since this process is tedious and expensive, computer simulations are very useful. Such a model needs to incorporate the relevant physical processes. A simulation system including real time user interaction and visualization together with efficient numerical techniques has been developed using an object oriented design.

Fig. 1. Turbine blade with cooling holes

1 Introduction

The efficiency of a gas turbine engine is influenced by the temperature at the inlet of the turbine. There is a growing tendency to use higher inlet temperatures which leads to an increase of the heat load on the turbine components. This heat load is caused by the exposure to an enormous heat flux from the burnt gas coming from the combustion chamber. To maintain acceptable safety standards, these components need to be protected against their severe thermal environment. Protection methods which have been introduced to prolongate the lifetime of turbine blades are coating, internal cooling and film cooling. We will focus on internal cooling. This cooling is achieved by compressing relatively cool air through holes in the longitudinal direction of the turbine blades. In order to increase the heat transfer in the holes, the wall of the cooling passage is provided with multiple ribs. These irregularities are called *turbulators*, after the turbulence they are supposed to cause in the flow. This way the cooling efficiency of these holes is improved in the sense that the amount of contact surface between metal and cooling air is increased and there is also a better heat exchange due to turbulence. It is known that the heat transfer is better than in the laminar case. Moreover less cooling air is needed.

One possibility for producing these holes is drilling. However, since the holes have such a complex shape and their diameter is only about a few millimeters, conventional
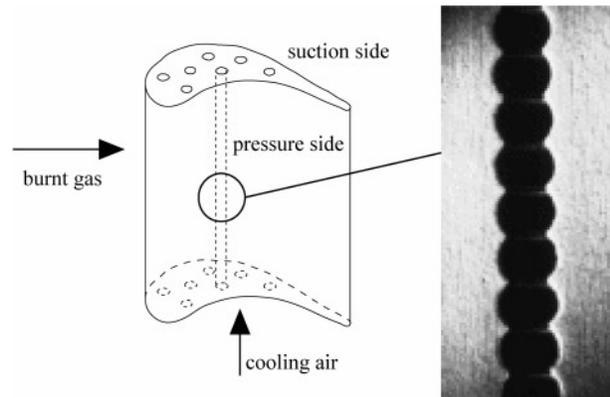
drilling techniques are not suitable. Therefore an *Electrochemical Drilling* (ECD) technique is used; i.e. an electrolytic process is employed where an anode moves gradually into the metal. Turbulated cooling holes are drilled by varying process parameters, such as speed, during drilling in a well defined and controlled way. The drilling of these holes requires a large number of well defined experiments on test pieces which are very expensive. Computer simulation of the ECD process may reduce the number of experiments. A model has been developed to determine the effect of parameter variations on the shape of the turbulators. Using the *Finite Element Method* (FEM) the dissolving rate of the metal is computed.

In Sect. 2 the relevant physical processes are discussed. Here we also derive the algorithm for computing the changing coordinates of the anode surface during passage of current. The boundary value problem is formulated in Sect. 3. The numerical techniques for solving this problem are addressed in Sect. 4. This involves space and time discretization. In Sect. 5 some results of simulations are presented.

An important issue in the modelling of physical processes is their real time characteristic and the ability to simulate it with a computer program. A software system able to simulate a physical process by integrating the computation, user interaction and visualization stages in a single real time framework has been developed and the ECD process
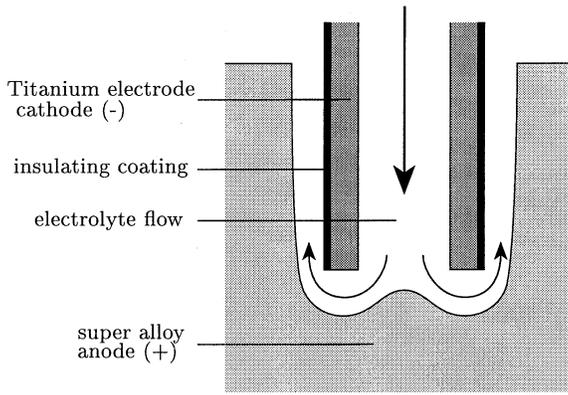
Correspondence to: wsanmn@win.tue.nl

Fig. 2. Electrochemical drilling

has been used as a case problem. In the ECD process, there are several physical parameters which have to vary in time in order to obtain a desired shape for the turbulators. The software simulation system allows the user to interactively control the variation of these parameters and to monitor the results of the simulation in real time.

This software system can be regarded as a general purpose numerical simulation environment offering direct user interaction with running FEM simulations. The system has been designed as an open environment capable of accepting several visualization metaphors and user interaction schemes as well as different numerical techniques. The use of an object oriented design and of C++ as implementation language have been proved to be valuable tools for the architecture of such a software project.

2 Physical modelling

The ECD process is based on electrolysis. The drill is a conducting cylinder with an insulating coating on the outside. This drill is lowered into the material with a certain speed s while a voltage U is applied to it. In this way a cylindrically shaped hole is obtained as illustrated in Fig. 2. Because of the axisymmetry we can essentially use two dimensional computational models. We assume that the electrode is already in the anode material. The coordinate system is chosen relative to the fixed anode. Another reasonable option might be choosing this system relative to the moving drill, since all the important physical processes take place near the tip of the electrode.

The corrosion of the anode surface is a direct result of electrolysis. This is a process where an electric potential difference is imposed on an anode and a cathode. The electrolyte, often a sulphuric acid, tends to corrode the anode surface in this electric field. The thus corroded material is removed by the electrolyte flow. To describe the process of electrolysis we use Faraday's law. In a global formulation this yields for the volumetric removal rate

$$\frac{dV}{dt} = \frac{e_a}{\rho_a} I, \tag{1}$$

where V is the *volume of the anode* (see also [7]), I the current, ρ_a the *density* and e_a the *electrochemical equivalent* of the anode. Both quantities are assumed constant, which is

realistic in a practical situation. The electrochemical equivalent is a material property defined by

$$e_a := \frac{A}{zF}, \tag{2}$$

where A denotes the *molar mass* of the anode metal, z the *number of electrons* exchanged in the anodic reaction and F *Faraday's constant*. In order to determine the recession rate, the current density flux has to be known. To this end the various physical processes involved in electrolysis are now considered.

– Transfer of electrical charge
The current which flows through the electrolyte is due to the movement of ions. The ions in the solution are the "charge-carriers" establishing the current, following the imposed field by the potential difference. Thus the potential gradient need not be equal to zero even if no current is flowing through the solution. We introduce the *electrolytic conductivity* κ_e. The *current density* $\mathbf{J}$ is a result of gradients in the *electric potential* ϕ and ion concentrations (see [10] p.23). Since we have electric neutrality we can state that the motion of the fluid, which is electrically neutral, does not contribute to the current density. Because the typical velocity of the ions is much larger than the flow velocity of the electrolyte, the momentum of the ions is hardly affected by the strong flow. For a system without concentration gradients, $\mathbf{J}$ is the result of gradients in ϕ only.

$$\mathbf{J} = -\kappa_e \operatorname{grad} \phi, \tag{3}$$

according to Ohm's law. If we have considerable concentration gradients, the potential difference between two points is not only given by the current density and the conductivity, but also by the concentration gradients. For the electric neutrality of the solution to be preserved, it is necessary that the total production of charge in any volume element be equal to zero. This justifies the assumption of a divergence free electric field

$$\operatorname{div} \mathbf{J} = 0. \tag{4}$$

For isothermal electrolyte solutions with constant ion concentrations, Eqs. (3) and (4) result in the Laplace equation for describing ϕ.

– Current density distribution
For the computation of the potential two effects are important. First of all we have the ohmic drop in the body of the electrode and second the potential drop between the inner potential of the electrode and the inner potential of the adhering electrolyte close to the electrode surface due to the electrode reaction. Eventually a equilibrium state for electrode potential is reached at a given total current. This effect is referred to as *electrode polarization* (see [10] p.75). Near the tip of the cathode, where the current density is high, the electrode gets polarized. In this area the streamlines are very dense due to the transport of ions. As result it is difficult to choose proper boundary conditions for the potential. Therefore we assume that the electrodes are perfectly conducting and moreover that they are approximately nonpolarizable. Then we can impose essential boundary conditions for ϕ at the electrode's surface.

– Heat transfer in electrolyte solutions
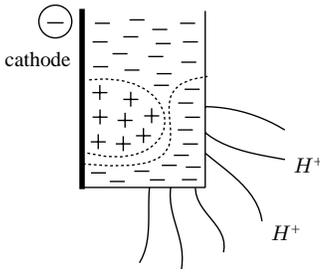For constant concentration gradients in the solution, the ionic

Fig. 3. Induced field by polarization at tip of cathode

Fig. 4. Conductivity of H_2SO_4 solution (17%)

mobility and hence the electrolyte conductivity, strongly depends on the temperature (see Fig. 4 from [6]). Instead of using the Laplace equation for describing the potential the conductivity has to be taken inhomogeneous, yielding the following equation

$$\text{div}(-\kappa_e(T)\,\text{grad}\,\phi) = 0. \tag{5}$$

The electrolytic conductivity depends implicitly on the *temperature* T. Gradients in the temperature of the solution are caused by the production of the heat Q due to passage of electric current through the solution. For an isothermal and homogeneous distribution of the concentrations the so-called *Joule heat* Q is defined by

$$Q := \frac{1}{\kappa_e}(\mathbf{J}, \mathbf{J}), \tag{6}$$

where $(\cdot\,,\cdot)$ denotes the inner product. This gives an acceptable approximation for Q only when gradients of temperature and concentrations are relatively small. If we do not take chemical reactions into account the distribution of temperature is described by

$$\rho_e c_p\Big(\frac{\partial T}{\partial t} + (\mathbf{v}, \text{grad}\,T)\Big) = -\lambda_e \Delta T + Q, \tag{7}$$

where $\mathbf{v}$ is the *velocity field* of the flowing solution according to pressure difference by a pump and λ_e the thermal conductivity. Besides the transfer of charge, the heat transfer is the most important effect in the ECD process. At high current densities, especially near the tip of the cathode, the solution temperature rises by the production of heat and the temperature gradients bring about the formation of gradients in the conductivity. But the pressure difference produced by the pump is so high, that the circulation of the solution takes place at high velocities. Moreover the solution always passes through a large reservoir such that it can be assumed that at the temperature of the solution going to the electrodes is almost constant (see also [11]). The increase of conductivity is therefore restricted to temperature boundary layers. If we want to model heat transfer we thus have to incorporate the motion of the fluid as well, for the sake of convection. This motion does not influence the transfer of electric charge. Since the appearance of conductivity gradients is restricted to boundary layers, we neglect this effect in our model.
– *Formation of gas*
Another effect, which leads to an inhomogeneous conductivity, is the evolution of gas. At the cathode H_2 is created due to reduction and at the anode O_2 as a result of oxidation. So the medium will consist of a mixture of solution and gas bubbles resulting in a decrease of the conductivity.
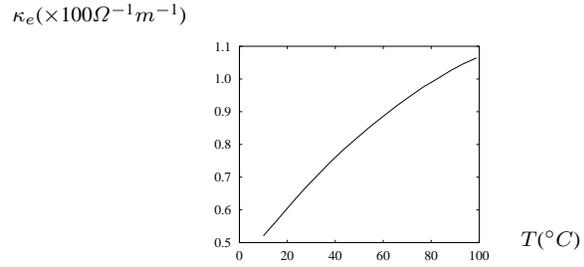
Again because of the strong flow this will only affect the conductivity locally and will therefore be neglected.

3 Mathematical equations

In the previous discussion about the relevant physical processes we have made some simplifying assumptions, which we will discuss below in a mathematical context. As a mathematical model for the electric potential ϕ in a homogeneous conducting solution without concentration gradients, we use the Laplace equation. Here it is assumed that the electrodes have such properties that we can impose Dirichlet conditions for the potential on the surface of the electrodes. The notion of time is actually present in Faraday's law only and not in the equation for the potential. Here we assume that the current density remains constant within a short time interval Δt, so that the shift of the anode surface along with the normal can be calculated from

$$\frac{d\mathbf{x}}{dt} = \frac{e_a}{\rho_a}(\mathbf{J}(\mathbf{x}), \mathbf{n})\mathbf{n}, \quad \mathbf{x} \in \delta\Omega_a. \tag{8}$$

This law relates the anode recession rate to the *current density flux*. Thus an expression is found for the instantaneous velocity by which the anode surface moves. In this formula $\delta\Omega_a$ denotes the *anode boundary* of the domain Ω (see Fig. 5).

We are mainly interested in the space between the electrodes, which is filled with an electrolyte and bounded by the electrodes and insulating walls. This space is connected by a channel with a reservoir. For the sake of simplicity it can be assumed that this channel is separated from the inter-electrode space by a fictitious insulating membrane (see Fig. 5). On this membrane we impose a homogeneous Neumann condition. Now the formulation for the electric potential ϕ yields

$$\begin{cases} \text{div}(-\kappa_e I\,\text{grad}\,\phi) = 0 & \text{in } \Omega \\ \phi = 0 & \text{on } \delta\Omega_a \\ \phi = -U & \text{on } \delta\Omega_c \quad (U > 0) \\ \dfrac{\partial\phi}{\partial\mathbf{n}} = 0 & \text{on } \delta\Omega\backslash(\delta\Omega_a \cup \delta\Omega_c) \end{cases} \tag{9}$$

Note that the homogeneous Neumann condition arises from the insulation of the electrode on the outside. The subscripts a and c refer to *the parts of the boundary at the anode and the cathode* respectively. Note that the time variable is treated explicitly because of Eq.(8). The imposed potential difference U is in fact a function of the time t. This parameter $U(t)$, together with drilling speed $s(t)$, constitute the process
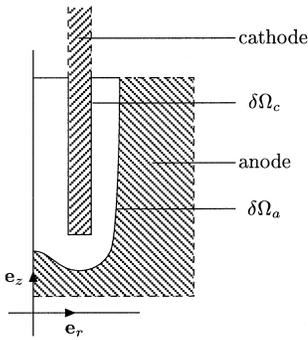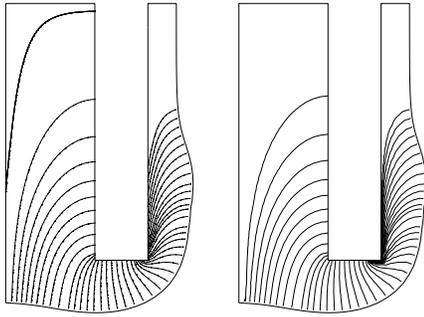
Fig. 5. Computational domain

(a) [FEM]   (b) [MHFEM]

Fig. 6. Particle tracking

parameters for manipulating the shape of the hole. Starting from an initial geometry the potential distribution can be computed. Then we can compute the current density on the anode boundary, which we use to compute the anode recession rate according to Faraday's law. The function s determines the position of $\delta\Omega_c$. For given functions s and U the problem is now well defined (see [3]). When these functions are constant one can give a asymptotic expansion for the gap width (see [8]).

4 Numerical approach

The mathematical model as derived in the previous section will be used for computer simulation. We discretize the equation using the *Finite Element Method* (FEM). If we would use conformal elements to discretize system (9), a discrete approximation for the potential ϕ can be found. The computed quantity can then be used to compute the current density flux field by taking the derivative numerically. Because the flux $\mathbf{J}$ is such an important variable in this process it should actually not be computed afterwards by post-processing for reasons of accuracy. Hence one should better use the *Mixed Hybrid Finite Element Method* (MHFEM). This solves the flux $\mathbf{J}$ implicitly, together with the potential, and also gives a better approximation for the flux. In this case the normal component of the flux will be continuous over the edges of each element.

In Fig. 6 we show the result of particle tracking starting from some user specified points on the boundary. These figures reveal that the achieved accuracy is different. Due to our mathematical model the gradients are singular near the tip of the cathode. This is a result of the fact we assumed

the electrodes to be nonpolarizable (see Sect. 2). Note that the insulating coating on the cathode has a certain thickness. The streamlines should wrap themselves around this coating. As one can see from Fig. 6 the MHFEM can handle these singularities numerically better than the conformal method.

Since our domain can deform rather awkwardly during the simulation we use triangular elements rather than quadrilaterals. This type of elements is easier to handle for mesh generation. As basis functions for the FEM on these elements we use piece-wise linear functions on the corners of the triangles to approximate the ϕ. The elements for MHFEM use midpoints only. In this case the potential is piece-wise constant per element and the flux linear and continuous over the edges of each triangle. It is clear that the MHFEM results in a larger system of equations than then the FEM. It can be shown that the mixed method is almost 2.5 times more expensive per iteration than the conforming method (see [4]). In our case the domain is expanding; hence especially for extensive simulations the number of elements can increase to quite a large number. To keep the computation time within reasonable limits we rather use the conforming method. To reduce memory usage iterative solvers are used. For this problem we use the *Preconjugated Gradient Method* with SSOR as preconditioner.

For notational purposes we write $t^i := i \cdot \Delta t$. Let Ω^0 be the initial geometry. Then the problem at $t = t^i$ can be formulated as follows:

$$\begin{cases} -\frac{1}{\kappa_e}\mathbf{J} + \operatorname{grad}\phi = 0 & \text{in } \Omega^i \\ \operatorname{div}\mathbf{J} = 0 & \text{in } \Omega^i \\ \phi = 0 & \text{on } \delta\Omega_a^i \\ \phi = -U(t^i) & \text{on } \delta\Omega_c^i \\ (\mathbf{J}, \mathbf{n}) = 0 & \text{on } \delta\Omega^i \setminus (\delta\Omega_a^i \cup \delta\Omega_c^i), \end{cases} \tag{10}$$

for the mixed method. After the spatial discretization the time has to be discretized. In order to obtain the new domain Ω^{i+1} the following Euler forward scheme is used:

$$\begin{aligned} \mathbf{x}^{i+1} &= \mathbf{x}^i + \Delta t \cdot \frac{e_a}{\rho_a}(\mathbf{J}(\mathbf{x}^i), \mathbf{n})\,\mathbf{n}, & \mathbf{x} \in \delta\Omega_a^i, \\ \mathbf{x}^{i+1} &= \mathbf{x}^i + \Delta t \cdot s(t^i)\mathbf{e}_z, & \mathbf{x} \in \delta\Omega_c^i, \end{aligned} \tag{11}$$

where the time step size Δt is taken constant. Implicit schemes would most likely allow larger time steps, but are difficult to implement for a moving mesh. This remeshing is performed almost every time step to preserve the quality of the mesh. As a result of this the topology changes every time. This is what prevents us from using previous solutions for implicit schemes. It also prevents us from using the previous solution as inital guess for the CG-solver.

5 Simulation results

In Fig. 7 the results of a simulation run are presented. At intervals of 50 secs the shape of the boundary is displayed. The computations have actually been carried out for the right half of the domain only, since we have a symmetric geometry (see Fig. 5). In this case the shape of the turbulators is not very pronounced. By changing the process parameters other shapes may be achieved. Also differently shaped electrodes produce different holes (see Fig. 8). This is not only due to the size of the electrode, but also due to the thickness
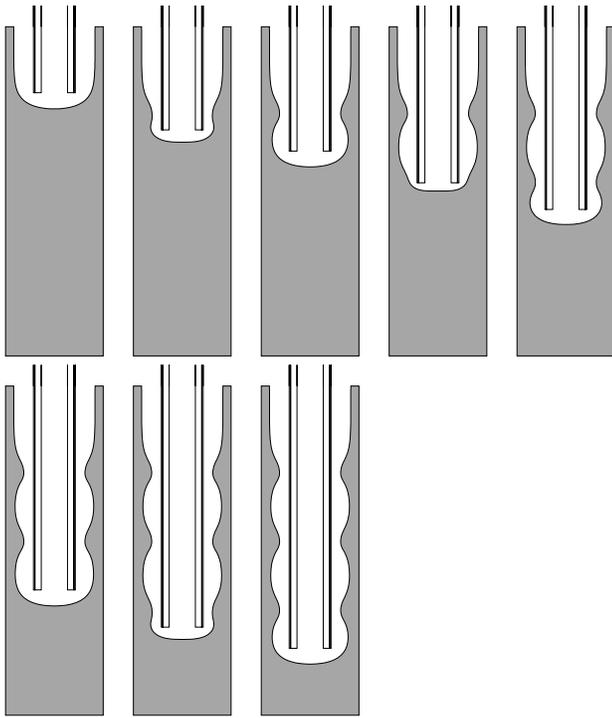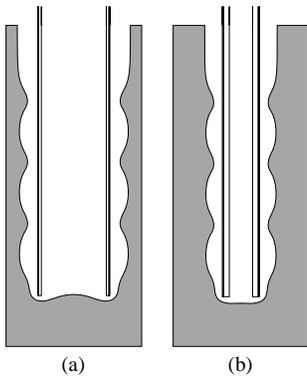
Fig. 7. Deformation of the anode surface

| (a) | (b) |

Fig. 8. Different types of electrodes

Fig. 9. Cooling duct with turbulators

of the insulating coating on the outside of the electrode. Instead of mirroring our 2D axisymmetric results we can also apply a rotation to the cross section and obtain a 3D body as illustrated in Fig. 9. The only way to validate our model is comparing the obtained geometry from a simulation run with an X-ray of a drilled hole in production (see Fig. 1). But even then, due to the complex shape, it is difficult to compare them. In order to demonstrate the validity of the model an experiment has been performed with step-wise variation of the potential difference. Its result is compared with the result of the simulation (see [1]).

6 Design of a real time simulation system

A particularly important feature of computer based simulations concerns the methods offered for interacting with the process and for retrieving data generated during the simulation. From this point of v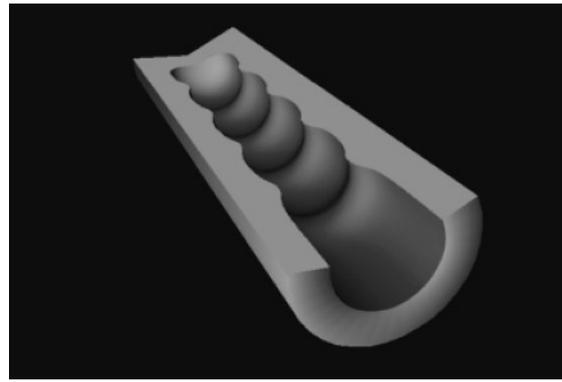iew one can distinguish between three large classes of simulation systems: non-interactive systems, interactive visualization systems and fully interactive systems (see [9]).

Non-interactive systems are the most common ones: the system operates as a pipeline having the problem definition phase (D) as the first step, followed by the numerical computations (problem solving) phase (C) and finally the result visualization phase (V). The three stages are loosely coupled. Generally they consist of separate applications communicating solely via files. The interactivity of such a system is practically inexistent; therefore the user is obliged to simulate a time dependent process by running the same pipeline over different sets of input data corresponding to different time instants.

Interactive visualization systems are also structured as a pipeline of the three phases previously described. The difference however, is that the coupling between consecutive stages is stronger now: the computational unit (C) can proceed solving a problem as soon as it is presented with some input data and the visualization unit (V) can start displaying a solution as soon as it is made available by the computational unit. By changing the input of the problem definition unit (D) in time, the user will produce and view a series of results corresponding to different time instants. The coupling between the problem definition phase and the computational phase is still weak and the user's possibilities for real time interaction are consequently quite limited.

Fully interactive systems group together the three phases of the simulation, allowing the user to interact with a running process and visualize the results of his interaction. Moreover, such a system is called *real time interactive* if the time elapsed between the user's interaction and the moment its result is visualized is short enough for the user to be able to continuously keep the process under direct control.

The system we have developed complies with the above requirements. The user can interact with a FEM problem during the entire duration of the process, from the problem definition moment during the end of the simulation. It is clear that we consider only time dependent processes. Note that this does not necessarily mean instationary problems. One may also think of a stationary problem whose definition is suddenly modified by user intervention, while the time derivatives can be neglected. The main feature of our system is therefore the ability of interacting with what we call a *running simulation* and to see the response of the

Fig. 10. Pipeline model for non-interactive or interactive visualization systems
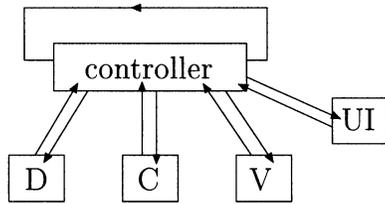

Fig. 11. Structure of the real time simulation system

system in real time. Interaction can come also in a different flavour since the system allows a process to determine its time evolution by itself. For example in the ECD process the solution obtained at a given time step will influence the initial conditions of the next time step. We can say that the system simulates a process that behaves autonomously. Another important feature is the system's ability to detect whether a change in the FEM problem's parameters caused by the user will induce several other changes in the system's state. For example, a user can move a point of the problem's definition domain and the system will automatically detect the need for re-meshing the domain, solving the problem on the new domain and displaying the new solution.

The structure of the simulation system is different from the pipeline model presented above. Rather than having several stages coupled in a unique fashion, we now have a set of independent modules that are managed and synchronized by a controller unit: problem definition module (D), computational module (C), visualization module (V) and user interface module (UI). The system's operation is event driven. The units can generate events (requests) to the controller and this one will respond by sending commands to the various modules that will carry on the requested tasks.

An object oriented design is used both for the interface part of the system and for its implementation in C++. The user can manipulate a FEM simulation in terms of high level objects as user points, curves, surfaces, domains and problems. These objects can be constructed, modified and deleted dynamically, allowing an interactive problem definition and also a flexible way of interacting with an existing problem. Several high level objects have been designed also for the visualization part: the user can create different cameras in which he can monitor running simulations by displaying their geometrical domains and scalar and vector fields. Cameras have standard methods for zooming and panning and facilities for displaying an object in various ways.

The user interface consists therefore of a set of C++ objects that can be assembled in order to define, modify, interact with and visualize a FEM problem. This allows for a quick and natural way of passing from an abstract problem definition to an interactive simulation of that process. It also allows re-usability of the problem definition building blocks since the user can construct a set of objects for a problem's definition and then employ them for several other problems.

```
double anod() {return  0;} // anode
double cath() {return -1;} // cathode
double cond() {return  1;} // conductivity

main()
{
    DOMAIN d(0.125); // domain with its gridsize

    USERPOINT p0(0,0),p1(1,0),p2(1,1),p3(0,1);

    USERLINE c0(p0,p1,d,anod,ESSENTIAL);
    USERLINE c1(p1,p2,d);
    USERLINE c2(p2,p3,d,cath,ESSENTIAL);
    USERLINE c3(p3,p0,d);

    SURFACE s(cond);
    s.addcurve(c0,PLUS); s.addcurve(c1,PLUS);
    s.addcurve(c2,PLUS); s.addcurve(c3,PLUS);

    s.compile(); d.addsurf(s); d.compile();

    PROBLEM pr(d); pr.settimestep(0.1);
                   pr.compgrad(TRUE);
                   pr.compile();

    CAMERA cam; cam.additem(d);
                cam.additem(pr,SHOW_SOLUTION);
                cam.additem(pr,SHOW_GRADIENT);
                cam.additem(pr,SHOW_STREAM);
                cam.reset();

    // read user interaction, compute and display
    for (;;) pr.solve(); // forever
}
```

The above program shows the definition and solving of problem (10) on a simplified domain. After defining the basic objects used to represent the problem, they are assembled by simple calls of C++ methods and then the problem can be solved by repeated calls of the problem object's solve method. The system automatically computes the solution, its gradient and stream function and continuously displays them in the camera. One can see that the problem definition code is not more complicated in structure than the usual problem description files used by several FEM packages. Besides a more modular approach, an object oriented interface allows a run time, dynamic definition. The user can start designing a computational domain, solve the problem on it and then possibly interactively changing the domain and having the system recalculate the solution.

Another example is the ECD simulation process (see Fig. 12). The user can interact with the running drilling simulation changing the drilling voltage and speed at any moment of the process by moving the respective sliders. The computational domain has been mirrored and displays different quantities in each cross section for each time instant. The user can also change the camera's viewing parameters during the simulation (focusing, for example, on the drill's tip while this one is continuously descending). Monitoring and interacting with the ECD process is very easy and intuitive in this way.

The system is implemented as an open software structure: objects can be added to the existing set in order to provide new functionality. Existing objects can be modified or new specializations of them can be provided in order to
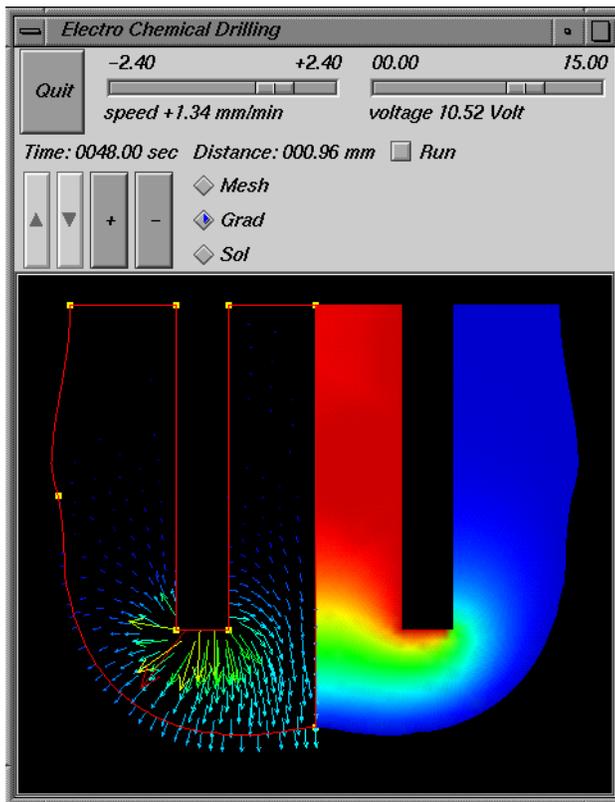
Fig. 12. Real time simulation with user interaction

change their functionality or to add new alternatives to the existing one. For example, new types of solver and preconditioner engine objects may be inserted within the system and selected dynamically at run time (see [2]). The user interface (implemented using the OSF/Motif and OpenGL libraries[1]) can be also enhanced in order to allow new visualization metaphors by creating different camera objects. To visualize the 2D computations in a 3D world AVS[2] has been used as a separate visualization tool (see Fig. 9).

References

1. Dijk, M.H.H.V., Verhappen, B.T.M., Jansen, J.K.M., Noot, M.J. FEM model to support electro chemical drilling of turbulated cooling holes. In: Proceedings of the International Gas Turbine and Aeroengine Congress and Exposition (The Hague, June 1994), no. 94-GT-305, ASME.

2. Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: elements of reusable object-oriented software. Addison-Wesley, Amsterdam, 1995.

3. Hardisty, H., Mileham, A.R., Shirvani, H. Finite element simulation of the electrochemical machining process. CIRP Annals **42**, 1 (1993), 201–204.

4. Kaasschieter, E.F. Mixed finite elements for accurate particle tracking in saturated groundwater flow. Advances in Water Resources **18** (1995), 277–294.

5. Kilgard, M.J. OpenGL and X, part 3: Integrating OpenGL with Motif. The X Journal (Jul/Aug 1994).

6. Lobo, V.M.M. (ed) Handbook of electrolyte solutions Part A, vol. 41 of Physical sciences data. Elsevier, Amsterdam, 1989.

7. McGeough, J.A. Principles of Electrochemical Machining. Chapman and Hall, London, 1974.

8. Noot, M.J. Numerical simulation of electrochemical drilling. In: Progress in Industrial Mathematics at ECMI 94 (Chichester, 1994), Neunzert, H. (ed), Wiley and Teubner, pp. 556–563.

9. Rosenblum, L., R. A. Earnshaw, J.E.E.A. (eds) Scientific visualization: advances and challenges. Academic Press, London, 1994.

10. Rousar, I., Micka, K., Kimla, A. Electrochemical Engineering I Parts A–C. Elsevier, Amsterdam, 1986.

11. Rousar, I., Micka, K., Kimla, A. Electrochemical Engineering II Parts D–F. Elsevier, Amsterdam, 1986.

---

[1] integration of both is described in detail in [5]

[2] Advanced Visual Systems Inc.